

**PATENT APPLICATION
DOCKET NO. 0100.0001150**

In the United States Patent and Trademark Office

5

FILING OF A UNITED STATES PATENT APPLICATION

Title:

METHOD AND APPARATUS FOR HIGH SPEED BLOCK MODE TRIANGLE RENDERING

10

Inventor:

Mark C. Fowler 5 Thayer Heights Road Hopkinton, MA 01748	Kevin M. Olson 155-11 Broadmeadow Road Marlborough, MA 01752
---	---

15

**Attorney of Record
John R. Garrett
Registration No. 27,888
P.O. Box 06229
Wacker Drive
Chicago, Illinois 60606-0229
Phone (312) 939-9800
Fax (312) 939-9828**

20

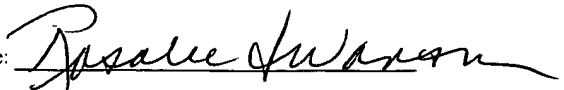
Express Mail Label No. **EL504284805US**

Date of Deposit: **August 1, 2000**

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addresses" service under 37 C.F.R. Section 1.10 on the 'Date of Deposit', indicated above, and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Name of Depositor: **Rosalie Swanson**
(print or type)

Signature:



25

5 METHOD AND APPARATUS FOR HIGH SPEED BLOCK MODE TRIANGLE RENDERING

Field Of The Invention

The invention relates in general to rendering triangles, and in particular to a method and apparatus for block mode rasterization with optimized edge detection.

10

Background Of The Invention

In the field of computer graphics, the graphics rendering pipeline is the core of real time graphics. The main function of the pipeline is to generate, or render, a two dimensional image, given a virtual camera, three dimensional objects, light sources, lighting models, textures, and more. The locations and shapes of the objects in the scene are determined by their geometry, placement of camera in the environment and the characteristics of that environment. The appearance of the objects is affected by material properties, light sources, textures, and lighting models.

20

In general, texturing is a technique for efficiently modeling the surface properties of an object. Objects or models are normally represented graphically by triangles, as these are the geometric primitives used by most graphics hardware. Color for each vertex of the surface is computed using location of the light sources and the properties, position, and the normal of the vertex and the properties of the material belonging to the vertex. Thus, the color is computed by taking into account lighting and the material, as well as the viewer's position. Texturing works in modifying the values used in the lighting equation. The pipeline uses complex mathematical equations to blend the colors, textures and other inputs to create properly colored pixels in the image.

30

In prior art graphics systems a texture memory contains a two dimensional representation of a texture to be mapped onto primitives. Cache provides temporary storage of portions of texture memory for improved access speed. Graphic primitives are stored in a primitives storage portion of memory, and define the size and shape of graphic elements, such as triangles. Primitives are processed by triangle set up modules and traversed using edge walker modules and span walker modules. A texture mapping engine performs the operation of mapping textures stored in a texture memory onto primitives. Pixel processing is performed by 3D pipeline and performs operations on the pixels and writes the resulting rendered image via a render backend module to a frame buffer. The image in the frame buffer is sent to a display.

A primitive typically represents some element of a graphic image on a display screen, and a texture map contains some graphical pattern or image that is to be mapped onto the primitive image. The edge walker and span walker are typically used in the prior art to map texture images onto primitives. A number of different techniques are known in the prior art for scanning primitives. For example, spans of horizontal pixel rows are defined, and each span has a unique y value representing the pixel row. Each span has two edges of the primitives to limit the extent of the span. The primitive is traversed span by span and for each span the pixels within the span are processed in left to right order. A texel in a texture map corresponding to the selected pixel is retrieved. Data for the retrieved texel is rendered for the selective pixel in the primitive. Other pixels in the primitive are processed similarly. A check is performed to determine whether the end of the current span has been reached. This check may be performed, for example, by comparing the x coordinate of the currently selected pixel with the edge of the primitive. If the end of the span has not been reached, the span walker continues by selecting the next pixel and repeating texel retrieval. Operations such as burst mode and pixel walk vectors increase the efficiency of the operation.

U.S. Patent No. 5,945,997 discloses a block and band oriented traversal and three dimensional triangle rendering. The disclosed system traverses and renders a graphic primitive by using block and band oriented traversal algorithms and texture mapping.

There is a need in the prior art to provide greater efficiency with a minimal amount of hardware for scanning primitives, and in particular, for texture mapping. All of the prior art methods attempt to find the edges of the primitive, by stepping outside the primitive as the pixels are edge walked or spanned.

Brief Description Of The Drawings

The features of the present invention which are believed to be novel, are set forth with particularity in the appended claims. The invention, together with further objects and advantages, may best be understood by reference to the following description taken in connection with the accompanying drawings, in the several figures of which like reference numerals identify like elements.

FIG. 1 depicts a 3D pipeline on a graphics chip according to the present invention.

FIG. 2 is a block diagram depicting the raster engine and pipeline depicted in FIG. 1.

FIG. 3 is a more detailed block diagram depicting the pixel blend stage depicted in FIG. 2.

FIG. 4 is a more detailed block diagram of the FIG. 3 pixel blend stage.

FIG. 5 is a block diagram of an individual math function stage in the FIG. 4 pixel blend stop.

Detailed Description Of a Preferred Embodiment of The Invention

FIG. 1 is a block diagram of a graphics chip 100 operatively coupled to a frame buffer 104 and receiving a command stream 101. In particular, the command stream 101 is received by graphics processor 103 from other stages in the computer equipment, which are not shown. The command stream contains the information for forming an image on a display. The graphics processor 103 processes the information as known in the art to provide primitives 120 that are representative of the desired image. The

primitives 120 are received by a setup engine 108 in a 3D pipeline 102. In the setup engine 108 the x y coordinates of the primitives 120 are transformed to form screen coordinates. The screen coordinates together with the z coordinates are also referred to as window coordinates. For example, a window can have a minimum corner at x_1, y_1 and a maximum corner at x_2, y_2 , where x_1 is less than x_2 , and y_1 is less than y_2 . The primitives are then mapped in this window and have x and y coordinates and z coordinates which indicate which primitives are in front of which primitives. These primitives are then passed on to the raster engine 110. The raster engine 110 is also referred to as a scan converter which converts the two dimensional vertices in screen space (with at least a z value, a color, and a texture coordinate associated with each vertex) into pixels. Unlike previous stages that perform polygon operations, the raster engine stage handles pixel operations. The pixel pipe 112 is operatively connected to the raster engine 110 and a render backend block 114 is operatively connected to the pixel pipe 112. A frame buffer 104, which is connected to the render backend block 114, has at least a color buffer 116 and a z buffer 118. The color buffer 116 stores color formation corresponding to pixels in the display frame, and the z buffer 118 stores corresponding z values for the pixels in the display frame.

Thus, in general the setup engine 108 produces primitive slope information based on the graphic primitives which is supplied to the raster engine 110. The raster engine 110 generates pixel fragments from the primitive slope information. Each pixel fragment includes the color value, a set of coordinates indicating a pixel position to which the value corresponds, and a z value for the fragment.

The raster engine 110 provides the pixel fragments to the pixel pipe 112. The pixel pipe 112 performs various operations that modify the color and texture of the pixel fragment as received from the raster engine 110. With regards to textures this is referred to as a texture mapping operation. The texture fragment resulting from the operations in the pixel pipe 112 then pass to the render backend block 114. The render backend block 114 blends the textured fragments with corresponding pixels in the frame buffer 104 as determined by the set of coordinates for each textured fragment. The z value for each

textured fragment is used to blend the fragment with the currently stored pixel information. When the operation is completed the frame buffer can be accessed by display hardware to obtain the pixel information for use in generating a display image.

5 FIG. 2 depicts a block diagram of a graphics system which utilizes the present invention. In a graphics system a processor 200 and a system memory 204 are coupled to core logic 202. Core logic 202 has a PCI bus 201, and an AGP bus 203 which couples the core logic 202 to the graphics module 206. The AGP bus 203 is operatively connected to the bus 210 in the graphics module 206. A graphics processor 208 controls
10 operation of a graphics module 206 that is coupled to the bus 210. The 3D pipeline 212 receives primitives from the bus 210 and outputs the graphic data to the render backend 214 which in turn provides the final pixels in the frame buffer 216. Display driver 218 reads the graphics data from the frame buffer 216 and displays the graphics image on the display 220.

15 A system for traversing and rendering a graphic primitive consists of a set up engine (such as set up engine 108 in FIG. 1) that outputs representative values of a graphic primitive. A raster engine (such as raster engine 110 in FIG. 1) receives the representative values of the graphic primitive and forms therefrom representative pixels.
20 The raster engine has at least a scan module that scans only pixels within the graphic primitive and assigns data values (such as color and texture values) to each of the pixels. The raster engine also has a look ahead module that identifies pixels that are inside the primitive. More specifically, the look ahead module determines if the next pixel in the y direction and in the x direction is inside the primitive. FIG. 3 depicts the scan module
25 302 coupled to the look ahead module 304 in the raster engine 300. In one embodiment, the scan module 302 is structured to perform block mode scanning, such as disclosed in U.S. Patent No. 5,945,997.

30 The graphic primitive can take the form of a triangle as depicted in FIGs. 4 and 5 of the drawings. As shown in FIG. 4, a primitive 400 has vertices at X0, Y0, X2, Y2 and X1Y1. The set up engine, such as set up engine 306 depicted in FIG. 3, determines the

edge functions E0, E1 and E2 for the triangular primitive 400. These edge functions are determined by the formulas depicted in FIG. 4 with the use of the three vertices, and the xy variables for a pixel. Given the x,y coordinates of a particular pixel, if the three edge functions provide positive results, this means that the pixel is within the primitive 400. If any one of the edge functions provides a negative result it indicates that the pixel is outside of the primitive 400. The set up engine 306 also determines which of the edges is longest for the primitive 400 (this being the E0 edge function). The set up engine 306 also determines the starting vertices XOY0 for the primitive.

As shown in FIG. 5, as pixels are scanned by the scan module 302 (such as pixels 501, 502 and 503), the look ahead 304 will calculate the edge functions for the pixel which is next in the y direction of the current pixel being processed. For example, referring to FIG. 5, while pixel 501 is being processed by the scan module 302, the look ahead module 304 calculates the edge functions for pixel 504 and concludes that this pixel is outside the triangle 500. When the scan module moves to the pixel 502, the look ahead module 304 then calculates the edge functions for the pixel 505 and will determine that the pixel 505 is within the triangular primitive 500. Having determined that the pixel 505 is within the triangular primitive 500 the look ahead module then stores the color or texture value associated with pixel 505 in a register. (This color or texture value associated with the pixel is also referred to in general as a data value.)

The operation of the scan module 302 and the look ahead module 304 is such that when the scan module 302 has processed the last pixel (such as pixel 503) in a scan line, the scanning of the next scan line will begin with pixel 505 which was identified by the look ahead module 304 as being inside the primitive 500. This eliminates what are referred to as dead cycles (that is, the processing of pixel 504 which is outside of the primitive 500) and therefore greatly increases the efficiency and speed, as well as reducing circuit complexity (as will be explained below).

Referring now to FIG. 6, the set up engine 600 provides slope information (such as, for example, for a color red), DR/DX and DR/DY. Registers 602 and 604 have inputs

connected to the set up engine 600 for storing the slope information for the x and y position, respectively. Outputs of the registers 602 and 604 are connected to inputs of a first multiplexer 606. An adder 608 has a first input 610 connected to the output of the first multiplexer 606. A third register 612 stores a characteristic value (such as a red value) for a predetermined pixel. A second multiplexer 614 has a first input 616
5 connected to the output 618 of the third register 612, and has an output 620 connected to a second input 622 of the adder 608. A third multiplexer 624 has a first input 626 connected to the set up engine 600 and a second input 628 connected to an output 630 to the adder 608. The output 632 of the third multiplexer 624 is connected to the input 634
10 of the register 612. These elements form the scan module described above.

A fourth multiplexer 640 has a first input 642 connected to the output 618 of the third register 612 and has a second input 644 connected to a second input 646 of the second multiplexer 614. The output 648 of the fourth multiplexer 640 is connected to the
15 input 650 of a fourth register 652. The output of the fourth register 652 is connected to the second input 646 of the second multiplexer 614. The fourth register 652 stores the data value for the pixel which is next in the y direction from the currently processed pixel, provided that the next pixel in the y direction is within the primitive. These elements form the look ahead module described above. The third register 612 stores the
20 data value for the current pixel being processed. The data value can be one of a color value and a texture value that is associated with the pixel.

Thus the present invention provides an efficient system and method for rendering triangles in a scan line, especially in view of reading texture data. By looking ahead to a
25 next pixel while processing a current pixel, the present invention overcomes the problem in the prior art wherein typical scan converter algorithms track edge functions for a pixel that is being currently processed. The present invention overcomes the problem in the prior art which wastes cycles by stepping outside the triangle when trying to determine if the next pixel is outside or within the triangle. The present invention improves the
30 texture operations by walking the triangle in a block mode rasterization pattern and implements an optimized edge detection algorithm that actually tracks the edge functions

